Evolutionary Optimisation of Evolving Connectionist Systems

Michael Watts and Nik Kasabov
Department of Information Science
University of Otago, PO Box 56
Dunedin
New Zealand
email: mike@kel.otago.ac.nz, nkasabov@otago.ac.nz

Abstract - The paper presents a method for optimising parameter values of evolving connectionist systems (ECoS) for life-long learning. The method is based on evolutionary computation principles and on genetic algorithms in particular. The method is illustrated on a spoken phoneme data classification task.

I. Introduction

Evolving Connectionist Systems (ECoS) are systems that evolve over time from incoming data streams through interaction with the environment, that is an ECoS adjusts its structure with a reference to the environment [6]. One of the main problems of ECoS is defining the optimal parameter values [7]. As the input stream is a continuous one often with changing characteristics parameter values may need to change in an on-line mode during the learning (evolving) process. The paper presents a method based on genetic algorithms (GA) for optimisation of ECoS parameter values. The paper is arranged as follows: Section II describes the general ECoS model, while Section III describes the Simple Evolving Connectionist System implementation of the ECoS paradigm. Section IV explains why it is necessary to optimise parameters of an ECoS network, and the Genetic Algorithm (GA) model used to optimise the SECoS is described in Section V. Experiments using the GA optimisation model and comparing it with unoptimised SECoS training are presented in Section VI. Conclusions are offered in Section VII.

II. Evolving Connectionist Systems

ECoS are multi-level, multi-modular structures where many modules have inter-, and intra-module connections. The evolving connectionist system does not have a clear multi-layer structure. It has a modular open structure. The functioning of the ECoS is based on the following general principles [6]:

1. ECoS learn quickly from a large amount of data through one-pass training
2. ECoS adapt in an on-line mode where new data is incrementally accommodated
3. ECoS have an "open" structure where new features (relevant to the task) can be introduced at any stage of the systems operation, for example, the system creates "on the fly" new inputs, new outputs, new modules and connections
4. ECoS memorise data exemplars for a further refinement, or for information retrieval
5. ECoS learn and improve through active interaction with other systems and with the environment in a multi-modular, hierarchical fashion
6. ECoS adequately represent space and time in their different scales; have parameters that represent short-term and long-term memory, age, forgetting, etc.

Several models of ECoS have been developed and applied on classification and time-series prediction tasks in a broad field of applications, such as: bioinformatics - gene expression data analysis; modeling the emergence of speech and language; mobile robot navigation; on-line image analysis; on-line adaptive process control; on-line financial time series prediction [8].

III. Simple Evolving Connectionist Systems

A. The Architecture of SECoS

The Simple Evolving Connectionist System, or SECoS, is a minimalist implementation of the ECoS principles [11]. It consists of three layers of neurons. The first is the input layer. The second, hidden layer is the layer which evolves. The activation of the nodes in this layer is determined as in Equation 1.

$$A_n = 1 - D \tag{1}$$

where A_n is the activation of the node n, and D is the distance between the incoming weight vector of n and the input vector I. Since D must be in the range $[0, 1]$,

the distance measure used in SECoS is the normalised distance between two vectors, as calculated according to Equation 2. Here N is the number of input nodes and W is the input to hidden layer connection weight matrix.

$$D_{n,I} = \frac{\sum_{i}^{N} \mid W_{i,n} - I_i \mid}{\sum_{i}^{N} \mid W_{i,n} + I_i \mid} \quad (2)$$

The third layer of neurons is the output layer. Here the activation is a simple multiply and sum operation over the hidden layer activations and the hidden to output layer connection weights. Saturated linear activation functions are applied to the hidden and output layers, and input values are expected to be normalised between 0 and 1.

B. The SECoS Learning Algorithm

The learning algorithm is as follows.

1. propagate the input vector I through the network
2. IF the maximum activation A_{max} of a node from the evolving hidden layer is less than the value of a parameter called sensitivity threshold S_{thr}
   - add a new hidden node
3. ELSE
   - evaluate the error between the components of the calculated output vector O_c and the desired output vector O_d
     - IF the error over the desired output is greater than another parameter value called error threshold E_{thr} OR the desired output node is not the most highly activated
       - add a node
     - ELSE
       - update the connections to the winning hidden node
4. repeat points 1,2 and 3 for each training vector.

When a node is added, its incoming connection weight vector is set to the input vector I, and its outgoing weight vector is set to the desired output vector O_d.

The incoming weights to the winning node j are modified according to Equation 3, where $W_{i,j}^t$ is the connection weight from input i to j at time t, η_1 is the learning rate one parameter, and I_i is the ith component of the input vector I.

$$W_{i,j}^{t+1} = W_{i,j}^t + \eta_1(I_i - W_{i,j}^t) \quad (3)$$

The outgoing weights from node j are modified according to Equation 4, where $W_{j,o}^t$ is the connection weight from j to output o at time t, η_2 is the learning rate two parameter, A_j is the activation of j, and E_o is the error at output node o.

$$W_{j,o}^{t+1} = W_{j,o}^t + \eta_2(A_j \times E_o) \quad (4)$$

ECoS create new nodes if new data vectors are presented to the system that are different from previously presented input vectors. In order to control the size of the ECoS additional operations are introduced, such as node aggregation. Nodes that are closer in the input space than a given threshold T1 and closer in the output space than another threshold T2 are aggregated so that a new node is created that accommodates the two close nodes. The connection weights of the new node in the input space W1 and in the output space W2 are formed as average values of the connection weights of the two aggregated nodes (see [7]).

IV. The Need to Optimise ECoS Networks

As discussed in [10], the size and performance of ECoS style networks are sensitive to the training parameters used. A high sensitivity threshold will cause neurons to be added for almost every training example, while a low error threshold will do the same thing. A high error threshold, on the other hand, will create a network that is too tolerant of errors, that is, a network that is not able to accurately classify examples that are presented to it. Setting the learning rates too high will cause the network to forget the previous training data much more quickly than is desirable, but if they are too low the network will not adapt. An aggregation threshold that is too high will destroy the captured knowledge, rendering the network almost useless, while a low aggregation threshold will allow the network to grow too large.

Training of an ECoS network is often a matter of finding the right trade-off between size of the network (in terms of the number of neurons in the evolving layer of the network) and the accuracy of the final network (how well it memorises the training data, and how well it generalises to new data). ECoS networks are deterministic systems: given the same ECoS network, the same training examples in the same order, and the same training parameters, the resulting trained network will be the same every time. To optimise the training of an ECoS network, therefore, requires optimising each of these three conditions. The first is considered beyond the scope of this paper: an attempt at optimising the initial state of an ECoS network using clustering techniques is described in [12]. This paper will therefore concentrate on the optimisation of the second two, the order of the training

examples and the training parameters.

Evolutionary algorithms [3], and in particular genetic algorithms (GA), [5] have been used several times in the past to optimise the training parameters of artificial neural networks (ANN) (for example, [1]). This paper will describe the use of similiar algorithms to optimise the training parameters of an ECoS style ANN, specifically a SECoS network (Section III).

V. GA Optimisation of ECoS

The GA used was broadly based on Goldberg's Simple Genetic Algorithm (SGA) [4], with the following modifications:

- real valued genes of type boolean, integer or floating point were used
- crossover was always performed, with mutation being performed on the child
- mutation was implemented as a re-initialisation of the gene
- one child per crossover was generated

Real valued parameters were used as they provide a more 'natural' fit to the problem [2]. The crossover and mutation procedures were regarded as being slightly more biologically plausible than those in Goldberg's SGA, as well as providing more genetic diversity. Mutation could also be implemented as a real numbered creep mutation, but this was felt to slow evolution too much. One child was generated for each crossover operation to increase the genetic diversity of the population.

The chromosomes used consisted of the training parameters, encoded as floating point numbers, followed by a long sequence of integers that described the order of the training set. That is, there was one integer gene for each example in the training data set, and the value of that gene specified which slot in the training data set that example should occupy.

For evaluation, each individual was decoded into a set of training parameters and a training data set, then a copy of the SECoS being optimised was trained and recalled with both the training and testing data sets. The error over the training and testing data sets, along with the change in size of the evolving layer of the SECoS, were used to calculate the fitness of the individual, according to Equation 5.

$$f_i = \frac{1}{1+e_t} + \frac{1}{1+e_r} + \frac{1}{1+\delta_n} \tag{5}$$

where:
f_i is the fitness of individual i,
e_t is the training error,
e_r is the error over the test set, and
δ_n is the normalised change in size of the evolving layers, as in Equation 6.

$$\delta_n = \frac{\mid n_{t+1} - n_t \mid}{d} \tag{6}$$

where:
n_t is the size of the evolving layer at time t,
d is the number of vectors in the training set.

The first two terms were intended to minimise the errors over both the training and testing data sets. The first of these was included to rate the networks memorisation ability, while the second rates the generalisation ability, as measured over previously unseen data. The error for each term was evaluated across the entire data set: thus, the order of the examples in the training set did not affect the error rate.

The third term was included to minimise the rate of change of the size of the network. Adding too many nodes would adversely affect the generalisation ability of the network, while removing too many (via aggregation) would damage its ability to remember training data.

VI. Experiments

The proposed algorithm was tested on the task of isolated phoneme recognition. This problem was chosen for several reasons: the problem is well characterised; copious amounts of data exist and are readily available; the data is complex, yet well understood. The data used was taken from the Otago Speech Corpus [9] (http://kel.otago.ac.nz/hyspeech/corpus/). This is a body of segmented words recorded from native speakers of New Zealand English, and covers all 45 phonemes present in the dialect. This corpus was used to create three sets of data A,B and C. Set A consists of utterances of forty three phonemes, taken from one male and one female speaker. Sets B and C consist of utterances of three phonemes (listed in Table I) taken from one additional male and one additional female speaker. The target phonemes were as in Table I, while the number of examples in each data set are listed in Table II.

ASCII Character	Example Word
/I/	p**i**t
/e/	p**e**t
/&/	p**a**t

TABLE I
EXEMPLAR PHONEMES

Each example consisted of a 78-element vector, which was made up of a three time step mel-scale transforma-

Data Set	Examples
A	10175
B	1040
C	519

TABLE II
NUMBER OF EXAMPLES IN EACH DATA SET

tion of the raw speech signal. The values were linearly normalised to lie between 0 and 1.

The experiments were carried out as follows: a SECoS network was trained for each phoneme on set A, using the training parameters in Table III. The trained network was then tested across all three data sets. The size of the evolving layer and accuracies of the initial trained networks are displayed in Table IV. These results show that the SECoS network has learned very well the training data set (set A) but has difficulty generalising to other speakers (given that there are only two speakers represented in set A, this is not really surprising). Adaptation of the networks to the two new speakers represented in sets A and B should allow the networks to better classify the new speakers.

Sensitivity Threshold	0.5
Error Threshold	0.1
Learning Rate One	0.5
Learnign Rate Two	0.5

TABLE III
SECOS TRAINING PARAMETERS

Data Set	/I/	/e/	/&/
A	100	97.2	97.9
B	36.8	50.9	62.2
C	32.2	32.2	76.8
Size	619	553	649

TABLE IV
INITIAL TRUE POSITIVE ACCURACIES AND SIZES

The GA training parameter optimisation algorithm described in Section V was therefore applied to each of the trained SECoS networks. Data set B was used as the training data, and the parameters used by the GA are as in Table V. The population size and number of generations were intentionally small: if this method is to be useful in real world applications, it must be able to run quickly, which means a small population and a low number of generations. The mutation rate of 0.01, and use of elitism and tournament selection were chosen to encourage a rapid evolution. After retrieving and testing the best network of the final generation, testing across all three data sets gave the accuracies shown in Table VI, where the sizes of the final networks is also displayed.

Population Size	20
Generations	20
Elitism	on
Mutation Rate	0.01
Selection	tournament

TABLE V
GA PARAMETERS

Data Set	/I/	/e/	/&/
A	97.9	97.1	83.9
B	98.6	100	94.7
C	58.6	70.9	88.1
Size	683	656	693

TABLE VI
TRUE POSITIVE ACCURACIES AND SIZES OF GA OPTIMISED SECOS

These results show that the networks all adapted well to the new data. The accuracies across data set B were all over 90%, while the accuracies over set C (consisting of further utterances of the same speakers sampled for set B) also improved. Although more than 1000 examples were used to adapt each network, less than 200 neurons were added to each network.

By way of comparison, the same initial networks were further trained on Set B using the same, unoptimised, parameters as before (Table III). The accuracies and sizes of these further trained networks are shown in Table VII.

Data Set	/I/	/e/	/&/
A	100	97.2	97.9
B	100	100	100
C	66.1	82.2	87.4
Size	864	896	952

TABLE VII
TRUE POSITIVE ACCURACIES AND SIZES OF FURTHER TRAINED SECOS

Although the accuracies over sets B and C were slightly

higher in some cases, many more neurons were added to each network, in some cases over 300. This shows that the GA is able to optimise the training parameters such that networks with competitive accuracies are created, while reducing the number of neurons added to the network during training.

VII. Conclusion

The paper presents a GA-based method for the optimisation of the parameters of simple evolving connectionist systems (SECoS). These parameters include sensitivity threshold, error threshold, learning rate one, and learning rate two, as well as the aggregation thresholds. The order of data presentation is also optimised.

The method has been shown to be able to determine parameters that result in SECoS networks that are significantly smaller than would otherwise be the case. Despite the smaller numbers of neurons, the accuracies of the optimised networks are competitive with the accuracies of networks trained using unoptimised parameters.

The high computational cost of this approach may limit its application, and in some cases may prevent it from being used in an on-line manner. The requirement that both a training and test data set be available to it may also cause problems. Research into ways in which an on-line data stream can be automatically partitioned into training and test sets is needed.

VIII. Acknowledgements

The paper is partially supported by a research grant UOOX0016 funded by the Foundation for Research, Science and Technology of New Zealand.

References

[1] B. Choi and K. Bluff. Genetic optimisation of control parameters of a neural network. In N. Kasabov and G. Coghill, editors, *Proceedings of ANNES'95*, pages 174–177, 1995.

[2] Lawrence Davis, editor. *Handbook of Genetic Algorithms*. International Thomson Computer Press, 1996.

[3] D. Fogel. *Evolutionary Computation*. IEEE Press, 1996.

[4] D.E. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, 1989.

[5] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.

[6] N. Kasabov. ECOS: A framework for evolving connectionist systems and the eco learning paradigm. In *Pro. of ICONIP'98, Kitakyushu, Japan, Oct. 1998*, pages 1222–1235. IOS Press, 1998.

[7] N. Kasabov. Evolving fuzzy neural networks for supervised/unsupervised on-line, knowledge-based learning. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 31(6):1220, 2001.

[8] N. Kasabov. *Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study, and Intelligent Machines*. Springer Verlag, London, 2002.

[9] S. Sinclair and C. Watson. The development of the Otago speech database. In N. Kasabov and G. Coghill, editors, *Proceedings of ANNES'95*. IEEE Computer Society Press, 1995.

[10] Michael Watts. An investigation of the properties of evolving fuzzy neural networks. In *Proceedings of ICONIP'99, November 1999, Perth, Australia*, pages 217–221, 1999.

[11] Michael Watts and Nik Kasabov. Simple evolving connectionist systems and experiments on isolated phoneme recognition. In Xin Yao and David B. Fogel, editors, *2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pages 232–239, 2000.

[12] B.J. Woodford and N.K. Kasabov. Ensembles of EFuNNs: An architecture for a multi module classifier. In *The proceedings of FUZZ-IEEE'2001. The 10th IEEE International Conference on Fuzzy Systems, December 2-5 2001, Melbourne, Australia.*, 2001.